Package: twoPhaseGAS (via r-universe)

May 27, 2026

Type Package

Title Two-Phase Genetic Association Study Design and Analysis with Missing Covariates by Design

Version 1.2.5

Description Provides functionality for designing and analysing two-phase genetic association studies. Phase 1 data usually come from genome-wide association study (GWAS) results and we assume phase 2 data will be part of a targeted genome sequencing or fine-mapping study. At design stage, the package assists in selecting a subset of individuals that will be sequenced for phase 2 via alternative approaches, including a flexible genetic algorithm (GA) for near-optimal designs. Once phase 2 data have been collected, the package implements methods to analyse phase 1 and phase 2 data together using semi-parametric regression models via the expectation-maximization (EM) algorithm. For more details see Espin-Garcia, Craiu and Bull (2018) <doi:10.1002/gepi.22099> and Espin-Garcia, Craiu and Bull (2021) <doi:10.1002/sim.9211>.

License GPL (>= 2)

URL https://github.com/egosv/twoPhaseGAS

BugReports https://github.com/egosv/twoPhaseGAS/issues

Depends R (>= 3.5.0)

Imports data.table, dfoptim, enrichwith, kofnGA (>= 1.2), MASS, Matrix, nloptr, parallel, stats, utils

Suggests gplots, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

Author Osvaldo Espin-Garcia [aut, cre], Apostolos Dimitromanolakis [aut], Radu V Craiu [ths], Shelley B Bull [ths]

Maintainer Osvaldo Espin-Garcia <oespinga@uwo.ca>

Repository https://cran.r-universe.dev

Date/Publication 2026-05-27 10:00:02 UTC

RemoteUrl https://github.com/cran/twoPhaseGAS

RemoteRef HEAD

RemoteSha 5f85bc43c06f44e49f413af5418eca64eb5bb4d9

Contents

DataGeneration_TPD 2
lrtest 3
optimTP.GA 5
optimTP.LM 8
print.lrtest.twoPhaseSPML 11
print.twoPhaseSPML 12
twoPhase 13
twoPhaseDesign 14
twoPhaseHeuristic 16
twoPhaseSPML 17
twoPhaseSPML.control 19

Index **20**

`DataGeneration_TPD` *This code a function that generates a sample Y, Z, G given some initial parameters.*

Description

This code a function that generates a sample Y, Z, G given some initial parameters.

Usage

```
DataGeneration_TPD(
  Beta0 = 2,
  Beta1 = 0.5,
  Disp = 1,
  N = 5000,
  LD.r = 0.75,
  P_g = 0.2,
  P_z = 0.3,
  family = gaussian(),
  tao = 2/5
)
```

Arguments

Beta0	intercept Default 2
Beta1	genetic effect Default 0.5
Disp	dispersion parameter, e.g., variance of the error term when family is gaussian(), the reciprocal of the shape parameter when family is Gamma() Default: 1
N	Phase 1 sample size, i.e. GWAS data (Default: 5000)
LD.r	linkage disequilibrium (r) between G and Z (Default: 0.75)
P_g	minor allele frequency for G, the causal SNP
P_z	minor allele frequency for Z, the GWAS SNP
family	member of the exponential family (see family;'quasi' models not available). Default gaussian().
tao	quantile value to define the stratification of the response variable, Y (only applicable when family is gaussian()) (default: 2/5)

Value

A dataframe with complete data Y, G, Z, S, where G and Z come from the same haplotype determined by P_g, P_z and LD.r; Y is generated from E[Y|G] = family$linkinv(Beta0 + Beta1 x G). S is a 3 level variable determined by Y, Beta1, Disp and tao (only returned when family is gaussian). the function iterates across generated datasets until Z and Y are associated at a suggestive genome wide threshold of p<=1e-05.

Examples

```
data = DataGeneration_TPD()
```

---

lrtest *Likelihood ratio test for* twoPhaseSPML *models*

Description

Compares two nested twoPhaseSPML models via a likelihood ratio test (LRT), mirroring the interface of lmtest::lrtest().

Usage

```
lrtest(object, object2, ...)

## S3 method for class 'twoPhaseSPML'
lrtest(object, object2, ...)
```

Arguments

`object`	a fitted object of class `"twoPhaseSPML"` representing the *smaller* (null / restricted) model.
`object2`	a fitted object of class `"twoPhaseSPML"` representing the *larger* (alternative / unrestricted) model. The two models must have been fitted to the same data and with the same `miscov`, `auxvar`, and `family`. The larger model must have at least as many parameters as the smaller one.
`...`	further arguments, currently unused.

Details

The test statistic is

$$LRT = -2\big(\ell_0 - \ell_1\big),$$

where ℓ_0 and ℓ_1 are the maximised log-likelihoods of the smaller and larger models, respectively. Under the null hypothesis that the additional parameters are zero, LRT follows a χ^2 distribution with degrees of freedom equal to the difference in the number of estimated regression coefficients between the two models.

The function determines model order automatically: whichever of `object` and `object2` has fewer regression coefficients is treated as the null model, regardless of the order in which they are supplied.

Value

An object of class `"lrtest.twoPhaseSPML"`, which is a `data.frame` with one row per model and the following columns:

`#Df` number of estimated regression coefficients.

`LogLik` maximised log-likelihood.

`Df` change in degrees of freedom relative to the previous row (`NA` for the first row).

`Chisq` LRT statistic (`NA` for the first row).

`Pr(>Chisq)` two-sided p-value from the χ^2 distribution (`NA` for the first row).

The object also carries a `"heading"` attribute (a character vector) and is printed via `print.lrtest.twoPhaseSPML`.

See Also

`twoPhaseSPML`, `print.lrtest.twoPhaseSPML`

Examples

```
data.table::setDTthreads(1)
data <- DataGeneration_TPD(N=200)
set.seed(1)
R <- rep(0L, nrow(data))
R[sample(nrow(data), 175)] <- 1L
data[R == 0L, c("G1")] <- NA

# Null model (no G1)
res_Ho <- twoPhaseSPML(
```

```
  formula = Y ~ Z,
  miscov  = ~ G1,
  auxvar  = ~ Z,
  data    = data
)

# Alternative model (with G1)
res_Ha <- twoPhaseSPML(
  formula = Y ~ Z + G1,
  miscov  = ~ G1,
  auxvar  = ~ Z,
  data    = data
)

lrtest(res_Ho, res_Ha)
```

---

optimTP.GA *Genetic-algorithm optimal phase 2 design for a two-phase genetic association study.*

Description

Uses a genetic algorithm (GA) to search over individual-level phase 2 selection indicators $R_i \in \{0, 1\}$ for the subset of size n that maximises the anticipated Fisher information for the regression parameters of interest. Wraps kofnGA. When a parallel cluster is supplied, multiple independent GA runs are executed in parallel and the best solution across all runs is returned.

Usage

```
optimTP.GA(
  formula,
  miscov,
  auxvar,
  family,
  n,
  data,
  beta,
  p_gz,
  disp = NULL,
  ga.popsize,
  ga.propelit,
  ga.proptourney,
  ga.ngen,
  ga.mutrate,
  ga.initpop = NULL,
  optimMeasure,
  K.idx = NULL,
```

```
  seed = 1,
  verbose = 0,
  cluster = NULL
)
```

Arguments

formula	A two-sided formula specifying the regression model of interest, e.g. Y ~ G + Z. All variables except those in miscov must be present in data.
miscov	Right-hand-side formula identifying the missing-by-design covariate(s) (the sequencing variant), e.g. ~ G.
auxvar	Right-hand-side formula identifying the auxiliary variable (the GWAS tag SNP), e.g. ~ Z. Must be present in data.
family	GLM family object (see family). Default: gaussian().
n	Integer. Target phase 2 sample size.
data	A data.frame containing the phase 1 data. Must include all variables in formula (except those in miscov) and auxvar.
beta	Numeric vector of assumed regression coefficients, one per term in formula (including the intercept). Setting beta[G] = 0 triggers the score-test (null-model) criterion; any non-zero value triggers the Wald (alternative-model) criterion.
p_gz	A data.frame with the assumed joint distribution of (miscov, auxvar). Must contain columns for each variable in miscov and auxvar and a column q of non-negative probability masses summing to 1. Rows with q <= 0 are silently removed.
disp	Numeric dispersion parameter (variance for Gaussian/Gamma; fixed at 1 for binomial/Poisson). If NULL (default), estimated as var(Y).
ga.popsize	Integer. GA population size (number of candidate solutions maintained per generation).
ga.propelit	Numeric in $(0, 1)$. Proportion of the population that is elite (carried forward unchanged to the next generation). The elite count is ceiling(ga.popsize * ga.propelit).
ga.proptourney	Numeric in $(0, 1)$. Proportion of the population used in each tournament selection. The tournament size is max(ceiling(ga.popsize * ga.proptourney), 2).
ga.ngen	Integer. Number of GA generations to run.
ga.mutrate	Numeric in $[0, 1]$. Per-index mutation probability. At each mutation step, each of the n selected indices is independently replaced by a randomly chosen non-selected index with probability ga.mutrate.
ga.initpop	Optional ga.popsize-by-n integer matrix of starting solutions (each row is a candidate subset of size n). If NULL (default), the initial population is generated randomly. When cluster is provided, the rows are split evenly across workers.
optimMeasure	Character string specifying the optimality criterion. One of "Par-spec" (parameter-specific), "A-opt" (A-optimality), or "D-opt" (D-optimality).

K.idx	Integer index (or vector of indices) identifying the parameter(s) of primary interest. Required when optimMeasure = "Par-spec".
seed	Integer random seed, set before GA initialisation. Default: 1.
verbose	Integer. If > 0, the best objective value is printed every verbose generations. Default: 0 (silent).
cluster	An optional parallel cluster object created by makeCluster. When provided, length(cluster) independent GA searches are run in parallel (each worker receives an equal share of ga.initpop rows, or a random initial population if ga.initpop = NULL), and the best solution across all workers is returned. The caller is responsible for starting and stopping the cluster. Default: NULL (sequential single-run GA).

Details

The fitness function evaluates the anticipated Fisher information matrix (FIM) for a candidate selection indicator R:

$$\mathrm{FIM}(R) = I_3/N + \sum_{i:R_i=1} (I_{2,i} - I_{3,i})/N,$$

where I_3 is the empirical score outer-product estimator (the numerically stable approximation to the full Louis (1982) observed FIM valid for large N, since $I_1 \approx I_2$ in the genotype expansion context), and $I_{2,i}$, $I_{3,i}$ are the per-individual complete-data score outer product and observed-data score outer product, respectively. Under H_0 (beta[G] = 0) the score-test Schur complement is optimised; under H_a the full inverse FIM is optimised.

Value

An object of class "kofnGA" (see kofnGA) with the following elements:

bestsol Integer vector of length n giving the row indices (in data) of the selected phase 2 individuals.

bestobj Numeric value of the optimality criterion at bestsol.

pop Final population matrix (ga.popsize rows, n columns), sorted by fitness.

obj Objective values for each row of pop.

old List with per-generation best, obj, and avg records.

time.elapsed Named numeric vector with the pre-processing (preproc) and GA (GA) times in seconds.

See Also

kofnGA, twoPhaseDesign, optimTP.LM

Examples

```
data <- DataGeneration_TPD(N = 500)
data <- data[, c("Y", "Z")]  # phase-1 data: no G column
p_gz <- data.frame(Z = c(0, 0, 0, 1, 1, 1, 2, 2, 2), # theoretical joint distribution
     G1 = c(0, 1, 2, 0, 1, 2, 0, 1, 2), # between G and Z (LD = 0.75, MAF_G = 0.2, MAF_Z = 0.3)
```

```
        q = c(0.486, 0.004, 0, 0.143, 0.276, 0.001, 0.011, 0.04, 0.039))

res_ga <- optimTP.GA(
  formula      = Y ~ Z + G1,
  miscov       = ~ G1,
  auxvar       = ~ Z,
  family       = gaussian(),
  n            = 100,
  data         = data,
  beta         = c(2, 0.5, 0.1),
  p_gz         = p_gz,
  ga.popsize   = 40,
  ga.propelit  = 0.1,
  ga.proptourney = 0.1,
  ga.ngen      = 50,
  ga.mutrate   = 0.01,
  optimMeasure = "Par-spec",
  K.idx        = 2L
)
cat("Best objective:", res_ga$bestobj, "\n")
cat("Selected indices (first 10):", head(res_ga$bestsol, 10), "\n")

## Parallel processing (2 workers)
cl <- parallel::makeCluster(2L)
res_par <- optimTP.GA(
  formula      = Y ~ Z + G1,
  miscov       = ~ G1,
  auxvar       = ~ Z,
  family       = gaussian(),
  n            = 100,
  data         = data,
  beta         = c(2, 0.5, 0.1),
  p_gz         = p_gz,
  ga.popsize   = 40,
  ga.propelit  = 0.1,
  ga.proptourney = 0.1,
  ga.ngen      = 50,
  ga.mutrate   = 0.01,
  optimMeasure = "Par-spec",
  K.idx        = 2L,
  cluster      = cl
)
parallel::stopCluster(cl)
cat("Best objective (parallel):", res_par$bestobj, "\n")
```

`optimTP.LM` *Lagrange-multiplier optimal phase 2 design for a two-phase genetic association study.*

Description

Computes stratum-specific sampling fractions that maximise the anticipated Fisher information for the regression parameters of interest subject to a fixed total phase 2 sample size. The optimisation is performed by the COBYLA algorithm (`nloptr`) with a derivative-free fallback (`dfoptim::hjkb`) if COBYLA fails to converge.

Usage

```
optimTP.LM(
  formula,
  miscov,
  auxvar,
  strata,
  family,
  n,
  data,
  beta,
  p_gz,
  disp = NULL,
  optimMeasure,
  K.idx = NULL,
  min.nk = NULL,
  logical.sub = NULL
)
```

Arguments

`formula`	A two-sided `formula` specifying the regression model of interest, e.g.`Y ~ G + Z`. All variables except `miscov` must be present in `data`.
`miscov`	Right-hand-side formula identifying the missing-by-design covariate(s) (the sequencing variant), e.g.`~ G`.
`auxvar`	Right-hand-side formula identifying the auxiliary variable (the GWAS tag SNP), e.g.`~ Z`. Must be present in `data`.
`strata`	Right-hand-side formula defining the stratification variables used to cross-tabulate `data`, e.g.`~ Z + S`. All variables must be present in `data`.
`family`	GLM family object (see `family`). Passed directly to the information-matrix calculations. Default: `gaussian()`.
`n`	Integer. Target phase 2 sample size (number of individuals to be sequenced).
`data`	A `data.frame` containing the phase 1 data. Must include all variables in `formula` (except those in `miscov`), `auxvar`, and `strata`.
`beta`	Named or unnamed numeric vector of assumed regression coefficients, with one element per term in `formula` (including the intercept). These define the operating point at which the Fisher information is evaluated. Setting `beta[G] = 0` triggers the score-test (null-model) criterion; any non-zero value triggers the Wald (alternative-model) criterion.

`p_gz` A `data.frame` with the assumed joint distribution of (`miscov`, `auxvar`). Must contain one column per variable in `miscov` and `auxvar` plus a column `q` holding non-negative probability masses that sum to 1. Rows with `q <= 0` are silently removed. Typically generated by `twoPhaseDesign` from `maf_G` and `LD`, or supplied directly by the user.

`disp` Numeric dispersion parameter. For Gaussian and Gamma families this is the variance (σ^2) or dispersion (ϕ), respectively; for binomial and Poisson it is fixed at 1. If `NULL` (default) it is estimated as `var(Y)`.

`optimMeasure` Character string specifying the optimality criterion. One of `"Par-spec"` (parameter-specific: minimise the asymptotic variance of `beta[K.idx]`), `"A-opt"` (A-optimality: minimise `trace(FIM^{-1})`), or `"D-opt"` (D-optimality: maximise `log det(FIM)`).

`K.idx` Integer index (or vector of indices) into `beta` identifying the parameter(s) of primary interest for `optimMeasure = "Par-spec"`. Ignored for A- and D-optimality. Required when `optimMeasure = "Par-spec"`.

`min.nk` Optional integer scalar or vector of length equal to the number of strata. Specifies the minimum number of individuals to sample from each stratum. If a scalar, the same minimum is applied to all strata. Default: `NULL` (no minimum, i.e. zero).

`logical.sub` Optional logical vector of length `nrow(data)`. If supplied, only the rows where `logical.sub = TRUE` are used in the information calculations (the full `data` is still returned). Useful for sensitivity analyses on subsets. Default: `NULL`.

Details

The function computes, for each stratum defined by `strata`, the stratum-specific sampling fraction $\pi_k = n_k/N_k$ that maximises the chosen information criterion. The optimisation is constrained so that (i) the total selected sample equals `n`, (ii) no stratum is over-sampled ($n_k \leq N_k$), and (iii) the `min.nk` lower bounds are respected.

The baseline Fisher information uses the empirical score outer-product estimator I_3/N (see `optimTP.GA` for details).

Value

A `data.frame` with one row per stratum (as defined by `strata`) and the following columns from `xtabs(strata, data)`:

(strata variables) The stratification variable values.

Freq Stratum size N_k.

prR_cond_optim Optimal conditional sampling probability n_k/n (proportional allocation weight).

convergence Convergence code from the solver (`0` = success; codes from `nloptr::cobyla` or `dfoptim::hjkb`).

See Also

`twoPhaseDesign`, `optimTP.GA`

Examples

```
data <- DataGeneration_TPD(N = 1000)
data <- data[, c("Y", "Z")]  # phase-1 data: no G column
p_gz <- data.frame(Z = c(0, 0, 0, 1, 1, 1, 2, 2, 2), # theoretical joint distribution
     G1 = c(0, 1, 2, 0, 1, 2, 0, 1, 2), # between G and Z (LD = 0.75, MAF_G = 0.2, MAF_Z = 0.3)
        q = c(0.486, 0.004, 0, 0.143, 0.276, 0.001, 0.011, 0.04, 0.039))

data$S <- as.numeric(cut(data$Y,
            breaks = quantile(data$Y, probs = c(0, 0.4, 0.8, 1)),
            include.lowest = TRUE))

res_lm <- optimTP.LM(
  formula     = Y ~ Z + G1,
  miscov      = ~ G1,
  auxvar      = ~ Z,
  strata      = ~ Z + S,
  family      = gaussian(),
  n           = 200,
  data        = data,
  beta        = c(2, 0.5, 0.1),
  p_gz        = p_gz,
  optimMeasure = "Par-spec",
  K.idx       = 2L
)
head(res_lm)
```

`print.lrtest.twoPhaseSPML` *Print method for* `lrtest.twoPhaseSPML` *objects*

Description

Print method for `lrtest.twoPhaseSPML` objects

Usage

```
## S3 method for class 'lrtest.twoPhaseSPML'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

`x`	an object of class `"lrtest.twoPhaseSPML"`, as returned by `lrtest.twoPhaseSPML`.
`digits`	integer; number of significant digits. Defaults to `max(3L, getOption("digits") - 3L)`.
`...`	further arguments passed to `printCoefmat`.

Value

x is returned invisibly.

See Also

lrtest.twoPhaseSPML

`print.twoPhaseSPML` *Print method for* `twoPhaseSPML` *objects*

Description

Prints a concise, glm-style summary of a fitted twoPhaseSPML model.

Usage

```
## S3 method for class 'twoPhaseSPML'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	an object of class "twoPhaseSPML", as returned by twoPhaseSPML.
digits	integer; minimum number of significant digits to print. Passed to printCoefmat. Defaults to max(3L, getOption("digits") - 3L).
...	further arguments passed to or from other methods (currently unused).

Details

Under the *null* model (miscov absent from formula), score statistics (Sobs and Sexp) are displayed. Under the *alternative* model, Wald statistics (Wobs and Wexp) together with standard errors and p-values for the missing-by-design covariate(s) are displayed. The coefficient table always shows the estimated regression coefficients; standard errors and p-values are shown only when var_theta is available (alternative model).

Value

x is returned invisibly.

See Also

twoPhaseSPML

Examples

```
data.table::setDTthreads(1)
data <- DataGeneration_TPD(N=200)
set.seed(1)
R <- rep(0, nrow(data))
R[sample(nrow(data), 175)] <- 1
data[R == 0, c("G1")] <- NA

res_Ho <- twoPhaseSPML(
  formula = Y ~ Z,
  miscov  = ~ G1,
  auxvar  = ~ Z,
  data   = data
)
print(res_Ho)
```

`twoPhase` *Internal function to optimize over a range of maf, LD and betas.*

Description

Internal function to optimize over a range of maf, LD and betas.

Usage

```
twoPhase(
  beta = c(1, 1, 1),
  maf_G = 0.1,
  LD = 0.3,
  data = NA,
  n2 = NA,
  design_formula = Y ~ G + Z,
  family = gaussian(),
  useGeneticAlgorithm = FALSE
)
```

Arguments

`beta`	Vector of betas (length of 3, corresponding to intercept, effect size for G, effect size for Z).
`maf_G`	Maf for G (numeric, ranging from 0 to 1).
`LD`	Correlation r betweek G and Z (numeric, ranging from -1 to 1, r value).
`data`	Data frame with Y and Z variables.
`n2`	Phase 2 sample size
`design_formula`	Formula for the regression model, default is Y ~ G + Z. Z is the GWAS SNP, G is the sequence variant. Y is outcome.

`family` Distrubution of the outcome (default: gaussian()). Familes available for glm can be used here. See help(stats::family) for examples.

`useGeneticAlgorithm` If TRUE, use genetic algorithm in addition to Lagrange multiplier approach (slower). Default: FALSE

Value

sth

Examples

```
data = twoPhase()
```

`twoPhaseDesign` *Compute sample allocations for a two phase study design.*

Description

Compute sample allocations for a two phase study design.

Usage

```
twoPhaseDesign(
  beta,
  maf_G,
  LD,
  data,
  n2,
  design_formula = Y ~ G + Z,
  family = gaussian(),
  S,
  perc_Y = c(1/5, 4/5),
  p_gz,
  design = c("RDS", "LM", "GA", "PPS", "BAL", "COM", "TZL"),
  ndraws = 10,
  optimCriterion = c("Par-spec", "A-opt", "D-opt"),
  overallMethod = c("med-max", "cumm")
)
```

Arguments

`beta` Vector of betas (length of 3, corresponding to intercept, effect size for G, effect size for Z).

`maf_G` minor allele frequency for G (numeric, ranging from 0 to 1). Numeric or vector of possible values.

LD	Correlation r betweek G and Z (numeric, ranging from -1 to 1, r value). Numeric or vector of possible values.
data	Data frame with Y and Z variables.
n2	Phase 2 sample size.
design_formula	Formula for the regression model, default is Y ~ G + Z, where Z is the GWAS SNP, G is the sequence variant. Y is outcome. Rename the variables in your data.frame to match Y and Z, G is the seq-SNP not present in the data.frame.
family	Distrubution of the outcome. Default: gaussian(). Familes available for glm can be used here. See help(stats::family) for examples.
S	stratification of the outcome Y. Optional. Should be a numeric vector with strata categories (e.g. 1 1 2 2 3 3). If present, its length must be equal to the number of rows in data. Default: NULL. Needed when Y does not render itself into strata, e.g. Gaussian, Poisson, Gamma.
perc_Y	vector of percentiles in increasing order for which the outcome Y will be stratified. Default: c(1/5,4/5). Only used when S is NA. Note that setting up S is strongly suggested.
p_gz	data frame with the joint distribution between G and Z. See examples for the right format. Default: NULL. If present, values of maf_G and LD are disregarded in the analysis.
design	string for the design to use for phase 2 sample selection. One of residual-dependent sampling ("RDS"), optimal as defined by Tao, Zheng and Lin (2019) ("TZL"), optimal via Lagrange multipliers ("LM"), optimal via genetic algorithm ("GA"), probability proportional to size ("PPS"), balanced ("BAL") or combined ("COM") allocations. Default: "RDS". See details for a more explanations.
ndraws	integer that determines the number of draws to examine when design is one of "pps", "bal" or "comb" and the design parameter combinations is greater than 1. Default: 10
optimCriterion	string denoting the optimality criterion used during the optimization. One of "Par-spec" (default), "A-opt" or "D-opt". For parameter-specific, A-optimality or D-optimality, respectively.
overallMethod	string denoting the method to select the overall design when multiple design parameters are given. One of "med-max" (default) or "cumm" for median-maximum and cummulative frequencies, respectively. Note that in this version strata are always defined in terms of Z and S, i.e. a joint design, future implementations may relax this by allowing for only S or only Z (marginal designs, outcome- or covariate-dependent, respectively)

Value

sth

`twoPhaseHeuristic` *Select samples for phase 2 under heuristic designs.*

Description

Select samples for phase 2 under heuristic designs.

Usage

```
twoPhaseHeuristic(
  design = c("pps", "bal", "comb"),
  ndraws = 1,
  data = NA,
  n2 = NA,
  family = gaussian(),
  S = NULL,
  perc_Y = c(1/5, 4/5)
)
```

Arguments

`design`	Heuristic design to use for phase 2 sample selection. One of probability proportional to size ("pps"), balanced ("bal") or combined ("comb") allocations. Default: "pps".
`ndraws`	Number of draws of the heuristic design to generate Default: 1.
`data`	Data frame with Y and Z variables.
`n2`	Phase 2 sample size.
`family`	Distrubution of the outcome. Default: gaussian(). Familes available for glm can be used here. See help(stats::family) for examples.
`S`	stratification of the outcome Y. Optional. Should be a numeric vector with strata categories (e.g. 1 1 2 2 3 3). If present, its length must be equal to the number of rows in data. Default: NULL. Needed when Y does not render itself into strata, e.g. Gaussian, Poisson, Gamma.
`perc_Y`	vector of percentiles in increasing order for which the outcome Y will be stratified. Default: c(1/5,4/5). Only used when S is NA. Note that setting up S is strongly suggested. Note that in this version strata are always defined in terms of Z and S, i.e. a joint design, future implementations may relax this by allowing for only S or only Z (marginal designs, outcome- or covariate-dependent, respectively)

Value

sth

`twoPhaseSPML` *Performs inference on two-phase studies data via semiparametric maximum likelihood.*

Description

Performs inference on two-phase studies data via semiparametric maximum likelihood.

Usage

```
twoPhaseSPML(
  formula,
  miscov,
  auxvar,
  family = gaussian,
  data,
  start.values = NULL,
  control = list()
)
```

Arguments

`formula`	regression formula, note that if it does not contain the missing-by-design variable, `miscov`, it will return results under the null hypothesis. Hypothesis testing corresponds to the score statistic. Otherwise, estimates and hypothesis testing occur under the alternative hypothesis leading to Wald statistics. All the elements in `formula` except `miscov` must be present in `data0` and `data1`.
`miscov`	right hand side formula with the missing-by-design covariate(s), i.e. the potential causal locus (loci). Must be present in `data1` but absent in `data0`.
`auxvar`	right hand side formula with the auxiliary variable(s), i.e. the GWAS SNP from phase 1. Must be present in `data0` and `data1`.
`family`	member of the exponential family (see `family`; 'quasi' models not available). Default `gaussian()`.
`data`	a data frame containing data from both phases. Phase 2 data is assumed to be the result of the `complete.cases` with both `miscov` and `auxvar`. Missing values in variables other than `miscov` are not allowed and may lead to unexpected errors.
`start.values`	a named list with initial values for the regression parameters (`betas`) and/or the joint distribution between `miscov` and `auxvar` (`q`). Defaults to `NULL`.
`control`	a list of parameters for controlling the fitting process, i.e., number or EM iterations, tolerance and whether a trace is requested, it is passed to `twoPhaseSPML.control`.

Details

The function implements the semiparametric maximum likelihood (SPML) estimator for two-phase genetic association studies described by Espin-Garcia et al. The EM algorithm alternates between

an E-step that computes observation weights based on the current estimate of the joint distribution of the missing covariate and the auxiliary variable, and an M-step that updates the regression coefficients and the non-parametric distribution via iteratively re-weighted least squares and a closed-form update, respectively.

When `miscov` is absent from `formula`, inference is performed under H_0 and score statistics are returned. When `miscov` is included in `formula`, inference is performed under H_a and Wald statistics are returned.

Value

An object of class `"twoPhaseSPML"`, which is a list containing:

theta named numeric vector of estimated regression coefficients.

qGZ data frame with the estimated joint distribution of `miscov` and `auxvar`.

dispersion estimated dispersion parameter (variance for Gaussian; 1 for other families).

iter integer, number of EM iterations performed.

ll log-likelihood at convergence.

df degrees of freedom for the test statistic.

Sobs (null model only) observed score test statistic.

Sexp (null model only) expected score test statistic.

var_theta (alternative model only) named numeric vector of marginal variances for `theta`.

Wobs (alternative model only) observed Wald test statistic.

Wexp (alternative model only) expected Wald test statistic.

qG data frame with the estimated marginal distribution of `miscov`.

formula the `formula` used.

miscov the `miscov` formula used.

family the `family` object used.

null.model logical, `TRUE` when the model was fitted under H_0.

Examples

```
data.table::setDTthreads(1)
data <- DataGeneration_TPD(N=200)
set.seed(1)
R <- rep(0, nrow(data))
R[sample(nrow(data), 175)] <- 1   # random phase 2 subsample of 500
data[R == 0, c("G1")] <- NA

# Null model (score test)
res_Ho <- twoPhaseSPML(
  formula = Y ~ Z,
  miscov  = ~ G1,
  auxvar  = ~ Z,
  data    = data
)
print(res_Ho)
```

```
# Alternative model (Wald test)
res_Ha <- twoPhaseSPML(
  formula = Y ~ Z + G1,
  miscov  = ~ G1,
  auxvar  = ~ Z,
  data    = data
)
print(res_Ha)
```

`twoPhaseSPML.control` *Auxiliary for Controlling the EM algorithm in twoPhaseSPML.*

Description

Auxiliary for Controlling the EM algorithm in twoPhaseSPML.

Usage

```
twoPhaseSPML.control(EMmaxit = 1000, EMtol = 1e-05, trace = FALSE)
```

Arguments

`EMmaxit`	integer giving the maximal number of iterations of the EM algorithm.
`EMtol`	numeric positive convergence tolerance for EM algorithm covnergence.
`trace`	logical `TRUE`, prints progress at each EM iteration. Defaults to `FALSE`.

Details

The function implements the semiparametric maximum likelihood (SPML)

Value

A list with components named as the arguments.

Index

DataGeneration_TPD, 2

family, *3*, *6*, *9*, *17*
formula, *6*, *9*

kofnGA, *5*, *7*

lrtest, 3
lrtest.twoPhaseSPML, *11, 12*

makeCluster, *7*

optimTP.GA, 5, *10*
optimTP.LM, *7*, 8

print.lrtest.twoPhaseSPML, *4*, 11
print.twoPhaseSPML, 12
printCoefmat, *11, 12*

twoPhase, 13
twoPhaseDesign, *7*, *10*, 14
twoPhaseHeuristic, 16
twoPhaseSPML, *3, 4*, *12*, 17
twoPhaseSPML.control, *17*, 19